AMENDMENTS TO THE SPECIFICATION

Replace the paragraph starting at page 8, line 18, with the following replacement paragraph:

Figure 1 is a block diagram illustrating a system configured for executing applications defined by XML documents based on application control parameters specified in XML control documents, and generating XML log documents for analysis according to an embodiment of the present invention. The system 10 includes an application server 12, a Web server 14, an XML document database 16, and an XML document registry 18. The XML document database 16 is configured for storing the XML documents that define the applications, and the XML control documents that specify application control parameters for execution of the executable voice applications. The XML document registry 18 is configured for storing XML documents that specify application state information for respective user sessions. The web server 14, for example an Apache server, receives HTTP requests from a client 5 via the Internet and forwards to the client HTML-based web pages dynamically generated by the application server 12. In particular, the application server 12 is configured for executing applications defined by stored XML documents, also referred to generally as web applications, in response to the HTML requests from the client. The application server 12 is implemented as a server executing a PHP hypertext processor with XML parsing and processing capabilities, available open source at ~~http://www.php.net~~ the PHP website ("php.net").

Replace the paragraph starting at page 10, line 6, with the following replacement paragraph:

The application server 12 also includes a runtime environment 24 for execution of the parsed XML documents. Figure 2 is a diagram illustrating in detail the application runtime environment 24 of Figure 1, emphasizing the use of XML control documents to define application context. The application runtime environment 24 executes the voice-enabled web applications 60a and 60b by first parsing an XML control document 62 that specifies the necessary application control parameters for execution of the voice-enabled web applications 60a and 60b. Hence, the XML control document 62 provides basic runtime control defaults for the application runtime environment 24. In addition, the application runtime environment 24 may parse a second XML control file ~~64~~ (e.g., 64a, 64b) that provides user-specific attributes overlying the basic control defaults specified in the XML control document 62. Hence, the application runtime environment 24 may parse a first XML control document 62 for basic runtime control defaults for the application runtime environment 24, followed by parsing of a second XML control document ~~64~~ (e.g., 64a, 64b) for user-specific attributes for a corresponding subscriber ~~66~~ (e.g., 66a, 66b). Hence, the application runtime environment 24 obtains basic runtime control defaults and user-specific attributes from stored XML control documents, enabling the stored XML control documents to provide context information to the application runtime environment 24. The context information is then used by the application runtime apartment 24 for execution of the application-defining XML documents.

Replace the paragraph starting at page 11, line 25, with the following replacement paragraph:

Figure 3 is a diagram illustrating a method of generating XML control documents and XML logging documents according to an embodiment of the present invention. The disclosed method can be implemented by a processor which executes instructions stored on a computer readable medium. Figure 3 also illustrates the usage of XML control documents for execution of voice-enabled web applications by the application runtime environment 24. The method begins in step ~~80~~ 180, where the application runtime environment 24 executes a master configuration XML document in the document database 16, that specifies all the necessary XML control documents needed for execution of voice-enabled web applications. As readily apparent from the foregoing, the master configuration XML document is generated by a user of the browser 5, using an XML document editor tool, to specify all necessary XML control documents 62. The application runtime environment 24 then loads each XML control document 62 specified in the master configuration XML document in step ~~82~~ 182 and parses the loaded XML control document to obtain the control parameters.

Replace the paragraph starting at page 12, line 16, with the following replacement paragraph:

The application runtime environment 24, upon loading the XML control documents in step ~~82~~ 182, obtains the necessary context information for execution of the application-defining XML documents in response to user requests. If desired, the application runtime environment 24 may also be configured for accessing user-specific attributes in step ~~84~~ 184: in this case, the

application runtime environment 24 accesses the corresponding user-specific XML control document 64 in response to receiving a request from the subscriber to the voice-enabled web application to obtain the user-specific context information. Hence, the XML control documents 62 can be considered a basic level of control information necessary for execution of a voice-enabled web application; the user-specific XML control documents 64 can be considered the next higher-level of control information necessary for execution of a voice-enabled web application for a corresponding subscriber, and the application-defining XML documents can be considered the highest level of abstraction with respect to execution of the voice-enabled web applications by the application runtime environment 24.

Replace the pargraph starting at page 13, line 10, with the following replacement paragraph:

During execution of the application-defining XML documents based on the XML control documents 62 and 64, the application runtime environment ~~22~~ 24 calls a logging function in the log module 50 in step ~~86~~ 186 in response to parsing a prescribed XML tag, within one of the application-defining XML documents, that specifies a logging operation. The log module 50 generates the XML log document 70 in step ~~88~~ 188 based on the log entry parameters that may be specified in the application-defining XML documents, and stores the log document 70 in the document database 16 in step ~~90~~ 190.

Replace the paragraph starting at Page 13, line 17 with the following replacement paragraph:

Figure 4B is a diagram illustrating a sample log file 70 generated by the log module 50 in step ~~88~~ 188. The log file 70 includes a log file tag 72 that specifies a name attribute 74 and an application attribute 76 that specifies the application having generated the log file 70 during execution by the application runtime environment 24. The log file ~~72~~ 70 also includes log entry tags 78a and 78b that specify a standard log entry 80a and an error log entry 80b, respectively. Each log entry ~~80~~ (e.g., 80a, 80b) includes a relevance tag ~~82~~ (e.g., 82a, 82b) that specifies a log element type (e.g., BILLING, TRACE, NOTIFY) for the corresponding log entry ~~80~~ (e.g., 80a, 80b) : in other words, the relevance tag 82a specifies the log element types BILLING and TRACE, indicating that the log entry 80a is relevant for billing information and trace information. Hence, the log file 70 can be parsed by the XML parser 20 during execution of an analysis routine configured for locating log entries related to billing information, trace information, or the like. Consequently, the analysis routine can quickly identify XML log documents 70 that are relevant for the analysis under consideration.

Replace the paragraph starting at page 14, line 1, with the following replacement paragraph:

The sample log file 70 also includes a log code XML tag 84 that specifies a prescribed log code based on a log code index. In particular, the log code may be referenced to a log code index that is used by the application runtime environment 24 to correlate the log code to the corresponding event. The sample log file 70 also includes additional tags ~~86~~ 86a, 86b (e.g.,

PARAMETERS, TEXT) that specify the log element data. Hence, the structure nature of the log file 70 in XML format enables the log data to be more highly structured, such that the log file 70 can be parsed during execution of an analysis application configured for reviewing log files for selected log data; the analysis application can determine whether data is relevant based on the corresponding relevance tags ~~82~~ (e.g., 82a, 82b) such that nonrelevant data can be ignored based on the corresponding XML tag. In addition, logs may be written for both individual subscriber sessions and overall application information. The logging tags are descriptive enough to understand the log information using any XML viewer, although a custom log parser can be written using standard XML parsing functions to gather and format the log information.